## Task 1

For address, I have chosen **Levenshtein distance**, which is a form of edit distance measure for 2 strings. I have chosen this as we need a good match between the entire sequence of the strings and not just parts of it.

For phone and cuisine, I have chosen **Jaro Winkler similarity**, which is a form of boosted Jaro measure. This seems to be a good choice since we wouldn't mind sub-sequences matching in the string for these 2 fields.

## Task 2

For combining the field similarities, I have given a weightage of 0.6 to address, 0.3 to phone and 0.1 to cuisine.

The reason for this is, we need the address to match the most. Example:

Bruno 240 E. 58th St. New York --- Felidia 243 E. 58th St. New York City

The above record turns out to be similar if the weight for address is low since we can see that "E. 58th St. New York" part is repeated exactly in both strings.

Phone is next, and is more important and given more weight that cuisine since the cuisine types can be referred to differently. Example "French Bistro" and "French" are most probably similar but keeping a high weight for that might not be a good idea.

Hence, the order is Address (0.6) > Phone (0.3) > Cuisine (0.1)

If the sum of all these is greater than 0.755, I output the pair as matching.

## Task 3

For task 3, RLTK library provides a way to hash on a particular column. They call it "Blocking". I hashed on the "Cuisine" since there would be only a certain number of categories. Hashing on address/phone doesn't make sense since there would be too many buckets and way too less comparisons.

After implementing blocking, The number of comparisons went down from 176423 to 4730. This is obviously a huge decrease. The similarity combination doesn't perform that well on these pairs since the cuisine categories are also widely varied. Example a restaurant with cuisine "French Bistro" and one with cuisine "French" do not match even though they are a match because they will hash to 2 different buckets.